CLAIMS

1. A method of replacing an implementation module used by a system, including the steps of:
   i) creating an interface module;
   ii) creating a plurality of proxy functions within the interface module corresponding to a plurality of functions within the implementation module;
   iii) tracking entries into and exits out of the implementation module by the system;
   iv) when the implementation module is to be replaced:
      a. the interface module blocking entry by the system into the implementation module; and
      b. when the number of entries correspond to the number of exits, replacing the implementation module;

   wherein the system uses the functions within the implementation module by calling the proxy functions and wherein some of the global variables of the implementation module are stored within the interface module.

2. A method as claimed in claim 1 wherein no state information of the implementation module is stored within the implementation module.

3. A method as claimed in claim 1 wherein the interface module blocks entry by the system into the implementation module only when it is safe to do so.

4. A method as claimed in claim 1 wherein the system is an operating system.

5. A method as claimed in claim 1 wherein the system is an application.

6. A method as claimed in claim 1 wherein the interface module performs step (iii).

7. A method as claimed in claim 6 wherein the tracking is performed using a reference counter.

8. A method as claimed in claim 6 wherein the tracking is performed using reference flags.

9. A method as claimed in claim 6 wherein the tracking is performed using reference counts and reference flags.

10. A method as claimed in claim 5 wherein the interface module is statically linked to the application.

11. A method as claimed in claim 5 wherein the interface module is dynamically linked to the application.

12. A method as claimed in claim 1 wherein the system includes a plurality of threads and at least some of the threads use the implementation module.

13. A method as claimed in claim 2 wherein some of the state information of the implementation module is stored on a heap.

14. A method as claimed in claim 1 wherein the implementation module is replaced with an updated version.

15. A method as claimed in claim 1 wherein the implementation module is replaced with a corrected version.

16. A method as claimed in claim 1 wherein each proxy function has the calling name of the corresponding function and the corresponding function is renamed.

17. A method of converting an implementation module, comprised of a plurality of functions, to a replaceable implementation module, including the steps of:

i) creating an interface module;

ii) creating a plurality of proxy functions, corresponding to the implementation functions, within the interface module wherein the calling name of each proxy function is the calling name of the corresponding implementation function; and

iii) moving some global variables from the implementation module to the interface module;

wherein the interface module is arranged for tracking the number of implementation functions in use, blocking calls to use the implementation functions when the

implementation module is to be replaced, and replacing the implementation module when no implementation functions are in use.

18. A method as claimed in claim 17 including the step of:
renaming the calling names of the implementation functions.

19. A method as claimed in claim 17 further including moving some global variables from the implementation module to another module.

20. A method as claimed in claim 17 wherein no global variables which hold state information are left within the implementation module

21. A method as claimed in claim 17 wherein the interface module blocks calls to use the implementation functions only when it is safe to do so.

22. An interface module for an implementation module, including:
i) a plurality of proxy functions corresponding to a plurality of functions within the implementation module;
ii) a tracking mechanism for recording the number of implementation functions in use;
iii) a blocking mechanism for blocking calls to the implementation functions when the module is to be replaced;
iv) a replacement mechanism for replacing the implementation module when no functions are in use; and
v) global variables for extraction from the implementation module.

23. A system for replacing an implementation module, including:
i) a memory which stores an implementation module comprised of a plurality of functions;
ii) a memory which stores an interface module comprised of global variables extracted from the implementation module and a plurality of proxy functions each arranged for executing a corresponding implementation function; and
iii) a processor arranged for (a) relaying calls to use an implementation function to a corresponding proxy function, (b) tracking the use of the implementation functions, (c) blocking calls to the implementation functions when the

implementation module is to be replaced, and (d) replacing the implementation module when no implementation functions are in use.

24. A method of replacing an implementation module used by a system, the method being performed with the aid of an interface module, and the method including the steps of: creating within the interface module a plurality of functions corresponding to a plurality of functions within the implementation module; tracking entries into and exits out of the implementation module by the system; when the implementation module is to be replaced: (a) blocking entry by the system into the implementation module by using the interface module and (b) replacing the implementation module when the number of entries correspond to the number of exits; causing the system to use the functions within the implementation module by calling the proxy functions; and storing within the interface module some of the of the global variables of the implementation module.

25. A method of converting an implementation module, comprised of a plurality of functions, to a replaceable implementation module, the method being performed with the aid of an interface module; the method including the steps of: creating within the interface module, a plurality of proxy functions corresponding to the implementation functions wherein the calling name of each proxy function is the calling name of the corresponding implementation function; moving some global variables from the implementation module to the interface module; tracking, with the interface module, the number of implementation functions in use, blocking, with the interface module, calls to use the implementation functions when the implementation module is to be replaced, and replacing, with the interface module, the implementation module when no implementation functions are in use.

26. A system for performing the method of claim 1.

27. A system for performing the method of claim 17.

28. A system for performing the method of claim 24.

29. A system for performing the method of claim 25.

30. A memory storing a program for causing a computer to perform the method of claim 1.

31. A memory storing a program for causing a computer to perform the method of claim 17.

32. A memory storing a program for causing a computer to perform the method of claim 24.

33. A memory storing a program for causing a computer to perform the method of claim 25.

34. Storage media storing a program for causing a computer to perform the method of claim 1.

35. Storage media storing a program for causing a computer to perform the method of claim 17.

36. Storage media storing a program for causing a computer to perform the method of claim 24.

37. Storage media storing a program for causing a computer to perform the method of claim for causing the memory of claim 25.

38. A binary file including an interface module and a replaceable implementation module created according to the method of claim 17.

39. A binary file including an interface module as claimed in claim 22.

40. A method comprising the step of supplying a computer with a program for causing the computer to perform the method of claim 1.

41. A method comprising the step of supplying a computer with a program for causing the computer to perform the method of claim 17.

42. A method comprising the step of supplying a computer with a program for causing the computer to perform the method of claim 24.

43. A method comprising the step of supplying a computer with a program for causing the computer to perform the method of claim 25.